Package ‘crosstalkr’

October 12, 2022

Title Analysis of Graph-Structured Data with a Focus on Protein-Protein Interaction Networks

Version 0.8.0

Description Provides a general framework for the identification of nodes that are functionally related to a set of seeds in graph structured data. In addition to being optimized for use with generic graphs, we also provides support to analyze protein-protein interactions networks from online repositories. For more details on core method, refer to Nibbe et al. (2010) <doi:10.1371/journal.pcbi.1000639>.

License GPL (>= 3)

biocViews

Imports rlang, stats, magrittr, withr, readr, dplyr, stringr, tidyr, tibble, igraph (>= 1.2.0), Matrix, ensembldb, foreach, doParallel, ggplot2, EnsDb.Hsapiens.v79, STRINGdb

Encoding UTF-8

RoxygenNote 7.1.2

Suggests tidygraph, ggraph, testthat (>= 2.0.0), knitr, rmarkdown

Config/testthat/edition 2

VignetteBuilder knitr

NeedsCompilation no

Author Davis Weaver [aut, cre] (0000-0003-3086-497X)

Maintainer Davis Weaver <davis.weaver@case.edu>

Repository CRAN

Date/Publication 2022-09-16 22:36:10 UTC

R topics documented:

  as_gene_symbol .................................................. 2
  bootstrap_null ................................................... 3
  check_crosstalk ................................................ 4
  compute_crosstalk ............................................. 5
as_gene_symbol

Convert from most other representations of gene name to gene.symbol

Description
Convert from most other representations of gene name to gene.symbol

Usage
as_gene_symbol(x, edb = NULL)

Arguments
x vector of ensemble.gene ids, ensemble.peptide ids, ensemble.transcript ids or entrez gene ids
edb ensemble database object

Value
vector of gene symbols
Examples

#1) from numeric formatted entrez id
as_gene_symbol(1956)
#2) from character formatted entrez id
as_gene_symbol("1956")
#3) from ensemble gene id
as_gene_symbol("ENSG00000146648")
#4) From a vector of entrez ids
as_gene_symbol(c("123", "1956", "2012"))

bootstrap_null  Bootstrap null distribution for significance testing

Description

This function will generate a bootstrapped null distribution to identify significant vertices in a PPI given a set of user-defined seed proteins. Bootstrapping is done by performing random walk with repeats repeatedly over "random" sets of seed proteins. Degree distribution of user-provided seeds is used to inform sampling.

Usage

bootstrap_null(
  seed_proteins,
  g,
  n = 1000,
  agg_int = 100,
  gamma = 0.6,
  eps = 1e-10,
  tmax = 1000,
  norm = TRUE,
  set_seed = NULL,
  cache = NULL,
  seed_name = NULL,
  ncores = 1
)

Arguments

seed_proteins  user defined seed proteins

Usage

bootstrap_null(
  seed_proteins,
  g,
  n = 1000,
  agg_int = 100,
  gamma = 0.6,
  eps = 1e-10,
  tmax = 1000,
  norm = TRUE,
  set_seed = NULL,
  cache = NULL,
  seed_name = NULL,
  ncores = 1
)
check_crosstalk

gamma restart probability
eps maximum allowed difference between the computed probabilities at the steady state
tmax the maximum number of iterations for the RWR
norm if True, w is normalized by dividing each value by the column sum
set_seed integer to set random number seed - for reproducibility
cache A filepath to a folder downloaded files should be stored
seed_name Name to give the cached ngull distribution - must be a character string
ncores Number of cores to use - defaults to 1. Significant speedup can be achieved by using multiple cores for computation.

Value
data frame containing mean/standard deviation for null distribution

Examples

g <- prep_biogrid()
bootstrap_null(seed_proteins = c("EGFR", "KRAS"), g = g, ncores = 1, n = 10)

Description
This function is a helper function for plot_ct that verifies the input is a valid output of compute_crosstalk

Usage

check_crosstalk(crosstalk_df)

Arguments

crosstalk_df a dataframe containing the results of compute_crosstalk

Value
message if not correct object type, null otherwise
**compute_crosstalk**

Identify proteins with a statistically significant relationship to user-provided seeds.

**Description**

`compute_crosstalk` returns a dataframe of proteins that are significantly associated with user-defined seed proteins. These identified "crosstalkers" can be combined with the user-defined seed proteins to identify functionally relevant subnetworks. Affinity scores for every protein in the network are calculated using a random-walk with repeats (`sparseRWR`). Significance is determined by comparing these affinity scores to a bootstrapped null distribution (see `bootstrap_null`). If using non-human PPI from string, refer to the stringdb documentation for how to specify proteins.

**Usage**

```r
calculate_crosstalk(
    seed_proteins, 
    g = NULL, 
    use_ppi = TRUE, 
    ppi = "stringdb", 
    species = "homo sapiens", 
    n = 1000, 
    gamma = 0.6, 
    eps = 1e-10, 
    tmax = 1000, 
    norm = TRUE, 
    set_seed, 
    cache = NULL, 
    min_score = 700, 
    seed_name = NULL, 
    ncores = 1, 
    significance_level = 0.95, 
    p_adjust = "bonferroni", 
    agg_int = 100
)
```

**Arguments**

- `seed_proteins`: user defined seed proteins
- `g`: igraph network object.
- `use_ppi`: should `g` be a protein-protein interaction network? If false, user must provide an igraph object in `g`.
- `ppi`: character string describing the ppi to use: currently only "stringdb" is supported.
- `species`: character string describing the species of interest. For a list of supported species, see `supported_species`. Non human species are only compatible with "stringdb".
n  number of random walks with repeats to create null distribution
gamma  restart probability
eps  maximum allowed difference between the computed probabilities at the steady state
tmax  the maximum number of iterations for the RWR
norm  if True, w is normalized by dividing each value by the column sum.
set_seed  integer to set random number seed - for reproducibility
cache  A filepath to a folder downloaded files should be stored
min_score  minimum connectivity score for each edge in the network.
seed_name  Name to give the cached ngull distribution - must be a character string
ncores  Number of cores to use - defaults to 1. Significant speedup can be achieved by using multiple cores for computation.
significance_level  user-defined significance level for hypothesis testing
p_adjust  adjustment method to correct for multiple hypothesis testing: defaults to "holm". see p.adjust.methods for other potential adjustment methods.
agg_int  number of runs before we need to aggregate the results - necessary to save memory. set at lower numbers to save even more memory.

Value

data frame containing affinity score, p-value, for all "crosstalkers" related to a given set of seeds

Examples

#1) easy to use for querying biological networks - n = 10000 is more appropriate for actual analyses
compute_crosstalk(c("EGFR", "KRAS"), n =10)

#2) Also works for any other kind of graph- just specify g (must be igraph formatted as of now)
g <- igraph::sample_gnp(n = 1000, p = 10/1000)
compute_crosstalk(c(1,3,5,8,10), g = g, use_ppi = FALSE, n = 100)

crosstalkr  crosstalkr: A package for the identification of functionally relevant subnetworks from high-dimensional omics data.

Description

crosstalkr provides a key user function, compute_crosstalk as well as several additional functions that assist in setup and visualization (under development).
crosstalkr functions

compute_crosstalk calculates affinity scores of all proteins in a network relative to user-provided seed proteins. Users can use the human interactome or provide a network represented as an igraph object.

sparseRWR performs random walk with restarts on a sparse matrix. Compared to dense matrix implementations, this should be extremely fast.

bootstrap_null Generates a null distribution based on n calls to sparseRWR

setup_init manages download and storage of interactome data to speed up future analysis

plot_ct allows users to visualize the subnetwork identified in compute_crosstalk. This function relies on the ggraph framework. Users are encouraged to use ggraph or other network visualization packages for more customized figures.

crosstalk_subgraph converts the output of compute_crosstalk to a tidygraph object containing only the identified nodes and their connections to the user-provided seed_proteins. This function also adds degree, degree_rank, and seed_label as attributes to the identified subgraph to assist in plotting.

crosstalk_subgraph

Helper function to generate subgraph from crosstalk_df output of compute_crosstalk

Description

Useful if the user wants to carry out further analysis or design custom visualizations.

Usage

`crosstalk_subgraph(crosstalk_df, g, seed_proteins)`

Arguments

- `crosstalk_df` a dataframe containing the results of compute_crosstalk
- `g` igraph network object.
- `seed_proteins` user defined seed proteins

Value

a tidygraph structure containing information about the crosstalkr subgraph

Examples

```r
## Not run:
ct_df <- compute_crosstalk(c("EGFR", "KRAS"))
g <- prep_biogrid()
crosstalk_subgraph(ct_df, g = g, seed_proteins = c("EGFR", "KRAS"))
## End(Not run)
```
detect_inputtype

Determine which format of gene is used to specify by user-defined seed proteins

Usage
detect_inputtype(x)

Arguments
x vector of gene symbols

Value
"gene_symbol", "entrez_id", "ensemble_id" or "other"

dist_calc

Internal function that computes the mean/stdev for each gene from a wide-format data frame.

Description
This function is called by the high-level function "bootstrap_null". Not expected to be used by end-users - we only export it so that environments inside foreach loops can find it.

Usage
dist_calc(df, seed_proteins)

Arguments
df : numeric vector
seed_proteins user defined seed proteins

Value
a data frame containing summary statistics for the computed null distribution
**ensembl_type**

*Determine if ensembl id is a Protein, gene, or transcript_id*

**Description**

Determine if ensembl id is a Protein, gene, or transcript_id.

**Usage**

`ensembl_type(x)`

**Arguments**

- `x` : vector or single gene symbol

**Value**

character: "PROTEINID", "GENEID", "TRANSCRIPTID"

---

**final_dist_calc**

*Internal function that computes the mean/stdev for each gene from a wide-format data frame.*

**Description**

This function is called by the high-level function "bootstrap_null".

**Usage**

`final_dist_calc(df_list)`

**Arguments**

- `df_list` : list of dataframes from foreach loop in bootstrap_null

**Value**

a dataframe
is_ensembl

Determine if a character vector contains ensembl gene_ids

Usage

is_ensembl(x)

Arguments

x vector or single gene symbol

Value

logical

is_entrez

Determine if a character vector contains entrez gene_ids

Usage

is_entrez(x)

Arguments

x vector or single gene symbol

Value

logical
**match_seeds**

Identify random sets of seeds with similar degree distribution to parent seed proteins

**Description**

This function will generate n character vectors of seeds to be passed to sparseRWR as part of the construction of a boostrapped null distribution for significance testing.

**Usage**

`match_seeds(g, seed_proteins, n, set_seed = NULL)`

**Arguments**

- `g`: igraph object representing the network under study. specified by "ppi" in bootstrap_null
- `seed_proteins`: user defined seed proteins
- `n`: number of random walks with repeats to create null distribution
- `set_seed`: integer to set random number seed - for reproducibility

**Value**

list of character vectors: randomly generated seed proteins with a similar degree distribution to parent seed proteins

**norm_colsum**

Function to normalize adjacency matrix by dividing each value by the colsum.

**Description**

Function to normalize adjacency matrix by dividing each value by the colsum.

**Usage**

`norm_colsum(w)`

**Arguments**

- `w`: The adjacency matrix of a given graph in sparse format - dgCMatrix

**Value**

input matrix, normalized by column sums
Examples

# 1) Normalize by column sum on a simple matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
norm_colsum(w)

Description

Convenience function for plotting crosstalkers - if you want to make more customized/dynamic figures, there are lots of packages that can facilitate that, including: visnetwork, ggraph, and even the base R plotting library

Usage

plot_ct(crosstalk_df, g, label_prop = 0.1, prop_keep = 0.4)

Arguments

crosstalk_df  a dataframe containing the results of compute_crosstalk
g  igraph network object.
label_prop    Proportion of nodes to label - based on degree
prop_keep     How many proteins do we want to keep in the visualization (as a proportion of total) - subsets on top x proteins ranked by affinity score

Value

NULL, draws the identified subgraph to device

Examples

## Not run:
ct_df <- compute_crosstalk(c("EGFR", "KRAS"))
g <- prep_biogrid()
plot_ct(ct_df, g = g)

## End(Not run)
**ppi_intersection**  
*Function to allow users to choose the intersection of stringdb and biogrid*

**Description**  
Function to allow users to choose the intersection of stringdb and biogrid

**Usage**  
```r  
ppi_intersection(cache = NULL, min_score = 0, edb = "default")  
```

**Arguments**
- `cache`  
  A filepath to a folder downloaded files should be stored
- `min_score`  
  minimum connectivity score for each edge in the network.
- `edb`  
  ensemble database object

**Value**  
igraph object corresponding to PPI following intersection

---

**ppi_union**  
*Function to allow users to choose the union of stringdb and biogrid*

**Description**  
Function to allow users to choose the union of stringdb and biogrid

**Usage**  
```r  
ppi_union(cache = NULL, min_score = 0, edb = "default")  
```

**Arguments**
- `cache`  
  A filepath to a folder downloaded files should be stored
- `min_score`  
  minimum connectivity score for each edge in the network.
- `edb`  
  ensemble database object

**Value**  
igraph object corresponding to PPI following union
prep_biogrid Prepare biogrid for use in analyses

Description
Prepare biogrid for use in analyses

Usage
prep_biogrid(cache = NULL)

Arguments
- cache A filepath to a folder downloaded files should be stored

Value
igraph object built from the adjacency matrix downloaded from thebiogrid.org.

prep_stringdb Prepare Stringdb for use in analyses

Description
Basically a wrapper around the get_graph method from the stringdb package

Usage
prep_stringdb(
cache = NULL,
edb = "default",
min_score = 0,
version = "11.5",
Species = "homo sapiens"
)

Arguments
- cache A filepath to a folder downloaded files should be stored
- edb ensemble database object
- min_score minimum connectivity score for each edge in the network.
- version stringdb version
- species species code either using latin species name or taxon id

Value
igraph object built from the adjacency matrix downloaded from stringdb.
sparseRWR

**Description**

This function borrows heavily from the RWR function in the RANKS package (cite here)

**Usage**

`sparseRWR(seed_proteins, w, gamma = 0.6, eps = 1e-10, tmax = 1000, norm = TRUE)`

**Arguments**

- `seed_proteins`: user defined seed proteins
- `w`: The adjacency matrix of a given graph in sparse format - dgCMatrix
- `gamma`: restart probability
- `eps`: maximum allowed difference between the computed probabilities at the steady state
- `tmax`: the maximum number of iterations for the RWR
- `norm`: if True, w is normalized by dividing each value by the column sum.

**Value**

numeric vector, affinity scores for all nodes in graph relative to provided seeds

**Examples**

```r
# 1) Run Random walk with restarts on a simple matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
sparseRWR(seed_proteins = c(1,3), w = w, norm = TRUE)

# 2) Works just as well on a sparse matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
w = Matrix::Matrix(w, sparse = TRUE)
sparseRWR(seed_proteins = c(1,4), w = w, norm = TRUE)

#3) Sample workflow for use with human protein-protein interaction network
g <- prep_biogrid()
w <- igraph::as_adjacency_matrix(g)
sparseRWR(seed_proteins = c("EGFR", "KRAS"), w = w, norm = TRUE)
```
### supported_species

returns a dataframe with information on supported species

**Usage**

```r
supported_species()
```

**Value**

dataframe

### to_taxon_id

helper to convert user-inputs to ncbi reference taxonomy.

**Description**

helper to convert user-inputs to ncbi reference taxonomy.

**Usage**

```r
to_taxon_id(species)
```

**Arguments**

- `species` user-inputted species

**Value**

string corresponding to taxon id
Index

as_gene_symbol, 2
bootstrap_null, 3
check_crosstalk, 4
calculate_crosstalk, 5
crosstalk_subgraph, 7
crosstalkr, 6
detect_inputtype, 8
dist_calc, 8
enssembl_type, 9
final_dist_calc, 9
is_ensembl, 10
is_entrez, 10
match_seeds, 11
norm_colsum, 11
plot_ct, 12
ppi_intersection, 13
ppi_union, 13
prep_biogrid, 14
prep_stringdb, 14
sparseRWR, 15
supported_species, 16
to_taxon_id, 16